Introduction à l’utilisation du logiciel Matlab

Fekak Fatima-Ezzahra
École nationale des Sciences Appliquées de Fès

Objectifs du cours

- Connaissance de l'environnement Matlab
- Commandes utiles, opérations élémentaires
- Structures vecteur et matrice
- Introduction à la programmation (structure de test, structure répétitive, boucle…)
- Fichier de commande et de fonctions
- Écriture de fonctions
- Génération de graphique sous Matlab

C’est quoi Matlab ?

MATrix LABoratory est un logiciel/langage de calcul numérique et de visualisation graphique

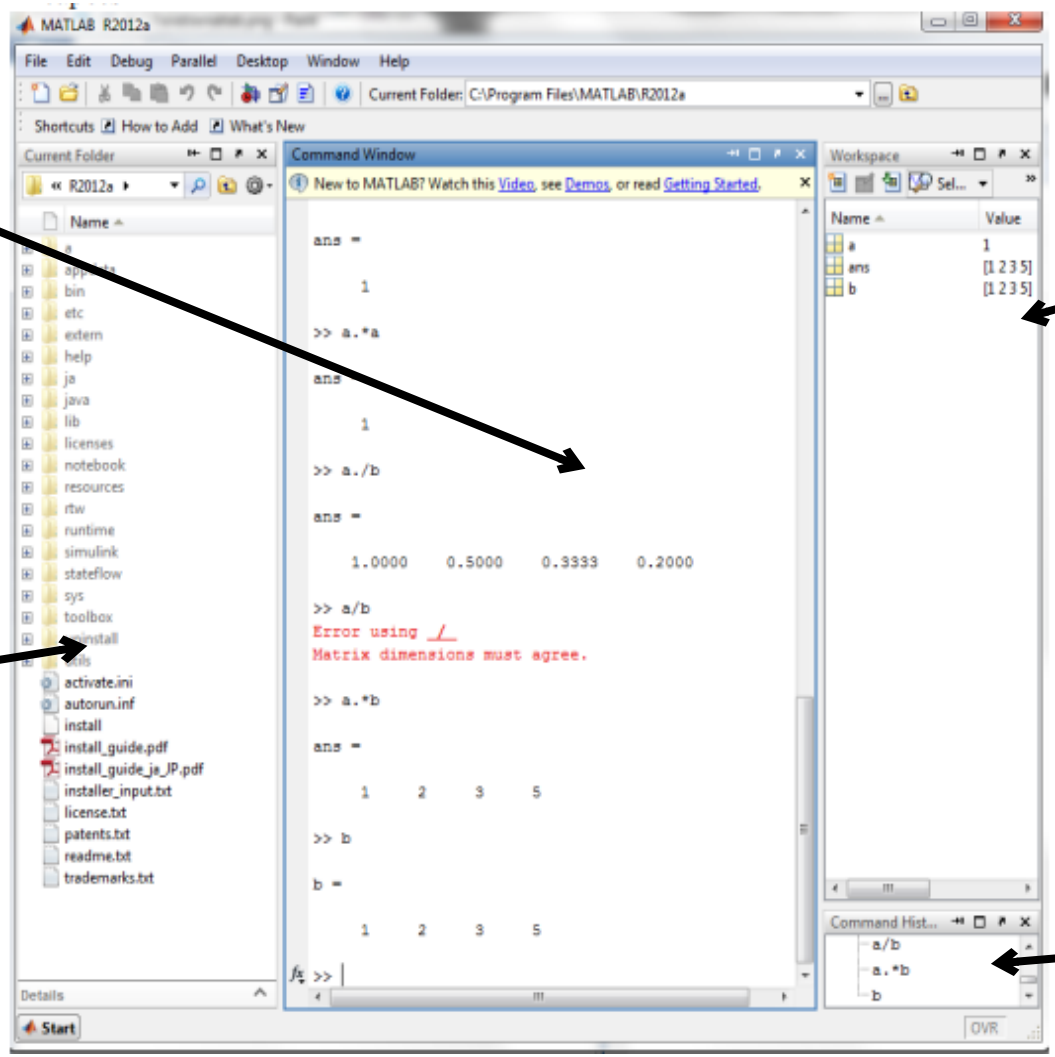
Exemples de domaines d’application :

- le calcul numérique dans le corps des réels ou des complexes ;
- le calcul de probabilités ou les statistiques ;
- le calcul intégral ou la dérivation ;
- le traitement du signal ;
- l'optimisation ;
- le traitement d'image ;
- l'automatisme.

L’environnement Matlab

Console: les commandes et leur résultats

Dossier de travail



Workspace: les données (nom, taille, type)

Historique: les commandes entrées dans la console

Les opérations élémentaires

Matlab comme simple calculatrice

Matlab peut bien sûr s'utiliser comme une simple calculatrice.

Exemples:

```
>> 1+2
```
donnera 3,

```
>> 4^2
```
retourne bien 16

```
>> (((1+2)^3)*4)/2
```
donne bien 54, puisque Matlab respecte les règles usuelles des mathématiques.

Matlab comme simple calculatrice

Définition des variables scalaires pour stocker un résultat:

```
>> a=1+2
>> b=2+3
>> a+b
```

retourne bien 8.

Calcul avec les variables complexes:

```
>> a=1+2i ;
>> b=2+3i ;
>> a*b
```

donne la réponse attendue, à savoir ans = -4 + 7i. Pour les complexes on peut utiliser i ou j avec: $i^2 = j^2 = -1$.

Remarque: terminer une ligne de commande par «;» permets de ne pas afficher le résultat

Quelques commandes utiles

Pour savoir les variables utilisées depuis le début de la session,
>> who
et pour en savoir plus sur elles
>> whos
Enfin, pour les détruire, on utilise
>> clear *variable*
ou encore pour toutes les variables
>> clear all
Certaines variables trés utiles sont déjà définies, comme
>> pi ou
>> ans qui retourne la dernière réponse donnée par Matlab

Remarque: en tapant ↑ on peut rappeler les commandes précédentes cela évite de taper plusieurs fois des commandes identiques

Les types de données dans Matlab

Les vecteurs dans Matlab

Définition d'un vecteur ligne v = (1 2 3 4 5):
>> v=[1 2 3 4 5]

Transposée du vecteur v:
>> v'

Définition d'un vecteur colonne
>> z=[1 ; 2 ; 3 ; 4 ; 5]

Remarque:
Pour obtenir z il est plus simple d'écrire:
>> z=v'
ou encore
>> z=[1 2 3 4 5]'

Les vecteurs dans Matlab

L’opérateur colon « : »
Ecrire « 1 :10 » signifie « tous les nombres de 1 a 10 » (c.à.d. 1 2 3 4 5 6 7 8 9 10

Ecrire « 1 :2 :10 » signifie « tous les nombres de 1 à10 avec un pas de 2 » (c.à.d. 3 5 7 9)

Remarque: Le pas est unitaire par défaut.

Cette notation doit être impérativement comprise car elle est utilisée sans arrêt. On s'amusera par exemple à écrire
>> 1 :10
>> 10 :-1 :1
>> 0 :pi/4 :pi
et à observer le résultat.

Les vecteurs dans Matlab

L’utilisation de opérateur colon « : » pour définir un nouveau vecteur
Exemple: vecteur ligne allant de 1 a 5 avec une incrémentation unité
>> w=[1 :1 :5]
ou encore
>> w=[1 :5]
La commande >> vec=[1 :2 :10]’ définie le vecteur colonne

$$\text{vec} = \begin{pmatrix} 1 \\ 3 \\ 5 \\ 7 \\ 9 \end{pmatrix}$$

Les vecteurs dans Matlab

Pour accéder a une valeur particulière d'un vecteur:
>> v(2) donne simplement accès au 2ème élément du vecteur v

Il est aussi possible d'utiliser les «:» pour sélectionner plusieurs éléments d'un vecteur, exemples:
>> v(1 :3) donne accès aux éléments qui vont de 1 a 3 (avec un pas de 1)

>> v(1 :2 :5) donne accès aux éléments qui vont de 1 a 5, mais en incrémentant avec un pas de 2.

>> v donne tout le vecteur est retourne, on obtient le même résultat par

>> v(:)

Les vecteurs dans Matlab

Quelques opérations vectorielles:
>> linspace(n,m,p) p nombres de n à m

>> lenght(x) longueur de x

>> x(i) ième coordonnée de x

>>x(i1:i2) coordonnées i1 à i2 de x

>>x(i1:i2)=[] supprimer les coordonnées i1 à i2 de x

>>[x,y] concaténer les vecteurs x et y

>>x*y' produit scalaire des vecteurs lignes x et y x'*y produit scalaire des vecteurs colonnes x et y

Les matrices dans Matlab

Il suffit d'ajouter une dimension aux vecteurs pour créer des matrices avec les mêmes notations:
>> A=[1 2 3 ; 2 4 5 ; 6 7 8 ;] génère la matrice

$$A= \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix}$$

Pour créer une matrice à partir de ses colonnes:
>> A=[[1 2 6]' [2 4 7]' [3 5 8]']

Pour lire les éléments, on utilise les même méthodes que les vecteurs:
>> A(1,3) permet d'accéder à l'élément (1,3) de la matrice A
>> A(1 :3,2 :3) retourne les lignes 1 à 3 et les colonnes 2 à 3
Le même résultat est obtenu par la notation
>> A(1 :1 :3,2 :1 :3)

Les matrices dans Matlab

Pour accéder à l'un des vecteur (ligne ou colonne) d'une matrice. Il faut alors se souvenir que l'on accède à toutes les valeurs d'un vecteur par la notation « : » et que, par conséquent, on écrira:

>> A(:,2) pour obtenir la deuxième colonne de A
>> A(:,3) pour la troisième colonne

De même
>>A(1, :) retourne toute la première ligne de la matrice A.

Pour avoir la transposée de la matrice A, on procède comme pour les vecteurs:
>> A‘

Les matrices dans Matlab

Commandes qui génèrent des matrices particulières :

>>rand(p) crée une matrice pxp dont les éléments sont aléatoires, distribues selon une loi de probabilité uniforme dans l'intervalle [0; 1[.

>>randn(p) crée une matrice pxp dont les éléments sont aléatoires, distribues selon une loi de probabilité gaussienne de moyenne nulle et de variance unité.

>>eye(p) crée une matrice identité pxp.

>>zeros(p) crée une matrice de zéros pxp.

>>ones(p) crée une matrices de uns pxp.

>>diag(x) crée la matrice dont la diagonale est le vecteur x

Remarque: Il est aussi possible d'utiliser ces commandes pour créer des matrices non carrées, en écrivant par exemple eye(5; 3).

Les matrices dans Matlab

Quelques opérations matricielles:

>> size(A) taille de A
>> rank(A) rang de A
>> inv(A) inverse de A
>> expm(A) exponentielle de A
>> det(A) déterminant de A
>> trace(A) trace de A
>> poly(A) polynôme caractéristique de A
>> eig(A) valeurs propres de A
>> [U,D]=eig(A) vecteurs propres et valeurs propres de A
>> A\b solution de Ax = b
>> b/A solution de xA = b

Les opérateurs matrices vecteurs

La multiplication de vecteurs et de matrices se fait en utilisant la notation « * ». Par exemple:

```
>> C=[1 3 4 ; 5 2 1 ; 3 1 2]
>> b=[3 1 2 ]
>> v=[1 3 4 ]
>> v*b'
>> v'*b
>> A*C
>> C*A
>> A*v’
```

Remarque: Bien évidemment, la multiplication n’est pas commutative.

Les opérateurs matrices vecteurs

La multiplication et la division terme par terme de vecteurs et de matrices se fait en utilisant la notation « .* » et « ./ ». Par exemple:

```
>> v.*b
>> v./b
```

De même pour les matrices:

```
>> A.*C
>> C./A
>> A.^2
```

Remarque: certaines fonctions, qui ne sont pas destinées directement aux matrices, s'appliquent directement aux éléments et n'ont donc pas besoin du point :

```
>> sin(v)
>> log(C)
>> exp(A)
>> 0.5-C
```

Quelques applications

1. Créer le vecteur v=(10 9.5 9 8.5 8)
2. Créer le vecteur w=(9 7 5 3 1)
3. Soit >>u=1:3 et >>v= [1 0 -1]
   Quel est le résultat >>u+v , >> u*v' et >>v'*u

4. Ecrire un mini programme permettant de résoudre le système suivant:

$$\begin{aligned} 2x + 3y + 4z &= 3 \\ x - y - z &= 0 \\ -x + 4y + z &= 5 \end{aligned}$$

Les chaînes de caractères

La chaîne de caractères est un vecteur ligne. Pour le créer, on entre les caractères en commencant et en terminant par ' (quote).

Exemple

>> ch='matlab‘

Question:

1. Trouver ch(4)?
2. Quelle commande faire pour écrire le mot Matlab ?

Les polynômes

Matlab représente un polynôme sous forme d'un vecteur ligne contenant les coefficients classés dans l'ordre des puissances décroissances.
Exemple: le polynôme P d'expression P(x) = x^2+ 6x + 9 est représenté par

>> P=[1 6 9]

Remarque: la commande >>roots(P) permet de trouver les racines du polynôme P.

Représentation graphique des résultats

Représentation graphique

Il existe plusieurs possibilités pour représenter un ensemble de points (x(i); y(i)). Les plus utilisées sont énumérées ci-dessous.

>> plot(x,y,'s') trace d'une courbe ou d'un nuage de points
>> bar(x,y,'s') trace sous forme d'un histogramme
>> stem(x,y,'s') diagramme en bâtons
>> stairs(x,y) trace en escalier des valeurs discrètes
>> fplot représente des fonctions
>> hist trace des histogrammes

's' est un paramètre facultatif constitue d'une chaîne de caractères qui spécifie le type de trace (couleur, différents tracés en pointilles, symboles pour le trace de points). Par défaut, le trace est continu. Tapez help plot pour avoir la liste des valeurs possibles pour 's'.

Représentation graphique

Quelques commandes pour la gestion de la fenêtre graphique:

>> axis([xmin xmax ymin ymax]) pour définir les échelles des axes
>> grid quadrillage du graphique
>> grid off
>> title('titre') titre pour le graphique
>> xlabel('titre') legende pour l'axe des abscisses
>> ylabel('titre') legende pour l'axe des ordonnees
>> legend('titre1','titre2',...) légende pour chaque courbe du graphique
>> text(x,y,'texte') texte explicatif à la position (x; y)
>> gtext('texte') texte positionnéàa l'aide de la souris

Introduction à la programmation

Opérateurs relationnels et logiques

1. Opérateurs relationnels:
- Inférieur: < (<=)
- Supérieur: > (>=)
- Égalité: ==
- Différent: ~=

2. Opérateurs logiques
- et: &
- Ou: |
- Non: ~ ou not

Les commandes structurées

L’instruction **for**:

for *variable = vecteur*
    instruction
end

Les colonnes du *vecteur* sont affectées, l'une après l'autre à la variable *variable* et pour chacune de ces valeurs, les *instruction* sont exécutées.

Remarque: il faut privilégier les operations vectorielles à l'utilisation de boucles.

Les commandes structurées

Exemple:
Ces quelques lignes calculent n! pour n = 1 a 100.

```
>> fact = zeros(1,100);
>> n=100;
>> fact(1)=1;
>> for k=2:n;
>> fact(k)=fact(k-1)*k;
>> end
```

Mais on peut aussi taper simplement : n=100; fact=cumprod(1:n);

Remarque: L'exécution du premier programme prend significativement plus de temps que l'exécution du deuxième programme.

Les instructions conditionnelles

L'instruction **if**:

if *conditions*

instruction

end

Ou:

if *conditions*

instruction

else

instruction

end

Ou:

Les instructions conditionnelles

Ou:

if *condition1*
instructions (exécutées si la *condition1* est vérifiée)
elseif *condition2*
instructions (exécutées si la *condition1* n'est pas vérifiée mais si la *condition2* l'est)
else
instructions (exécutées si ni la *condition1* ni la *condition2* ne sont vérifiées)
end

Les instructions conditionnelles

Exemple:
Le programme suivant simule le lancer d'une pièce.

```
p=0.5;
u=rand;
if u<p
disp('pile');
else
disp('face');
end
```

Les instructions conditionnelles

L’instruction **while**:

Ce format de boucle permet de s'arrêter conditionnellement (et non plus à rang fixe, comme dans une boucle **for**).

```
while conditions
    instruction
end
```

Les instructions conditionnelles

Application:

Construire la matrice tridiagonale A_1 d'ordre n:

$$\mathbb{A}_1 = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}, \text{ avec } h = \frac{1}{(n+1)^2}.$$

Les instructions conditionnelles

Réponse:

Construire la matrice tridiagonale A_1 d'ordre n:

```
A1 = sparse(n,n);
for i = 1:n
    A1(i,i) = 2;
end
for i = 2:n
    A1(i,i-1) = -1;
end
for i = 1:n-1
    A1(i,i+1) = -1;
end
A1 = (n+1)^2*A1;
```

Utilisation des fichiers dans Matlab

Sauvegarde de la fenêtre de commande

La commande **diary** *nomfichier* crée un fichier (journal de bord) intitulé *nomfichier* qui garde la trace de toutes les commandes que vous avez tapées dans la fenêtre de commandes ainsi que les réponses de l'ordinateur.

La commande **diary off** permet d'arrêter l’écriture du journal de bord

La commande **diary on** permet de la reprendre.

Sauvegarde des données Matlab

>> save *nomfichier var1 … varn* sauve la valeur des variables de var1 à varn dans un fichier **binaire** *nomfichier*.mat

>> save *nomfichier* sauve l'ensemble des variables existant dans l'espace debtravail de Matlab dans un fichier binaire *nomfichier*.mat

>> save -ASCII *nomfichier var1… varn* sauve la valeur des variables de *var1* à *varn* dans un fichier texte *nomfichier*

>> load *nomfichier* permet de récupérer toutes les données sauvegardées dans le fichier *nomfichier*.mat

Les fichiers d’instruction (scripts)

Ce sont des fichiers texte avec une extension .m
Ils contiennent des suites d'instructions Matlab qui sont exécutées les unes après les autres.

Exemple:
Sauvez dans le répertoire courant les lignes suivantes sous le nom losange.m :

```
x=[0 -1 0 1 ; -1 0 1 0 ];
y=[-1 0 1 0 ; 0 1 0 -1];
plot(x,y)
```

La commande losange tracera un losange. Si ce fichier est placé dans un répertoire accessible, la commande losange devient une commande Matlab comme toutes les autres.

Les fichiers de fonctions

Comme les fichiers d'instruction, ce sont des fichiers texte avec une extension .m
Leur syntaxe est **particulière**. Ils contiennent la définition d'une fonction et portent le nom de cette fonction.

Exemple 1:

```
function d=densnorm(x)
% densnorm : densite de la loi N(0,1)
% densnorm(x) retourne (1/sqrt(2*pi))*exp(-x^2/2)
d=(1/sqrt(2*pi))*exp((-x.^2)/2);
```

Si ce fichier est place dans un répertoire accessible, la fonction densnorm devient une fonction Matlab comme toutes les autres.
Le texte placé en commentaire est le contenu de l'aide pour la nouvelle fonction.

Les fichiers de fonctions

Fonction à plusieurs arguments: si plusieurs arguments sont fournis, on doit les séparer par une virgule ; et si plusieurs sont retournes, il faut les mettre entres crochets

Exemple:

```
function [d,dx,dy]=distance(x1,y1,x2,y2)
d=sqrt((x2-x1)^2 + (y2-y1)^2)
dx=x2-x1
dy=y2-y1
```

permet de retourner la distance absolue et les distance en x et y entre deux points de coordonnées (x1; y1) et (x2; y2).
En tapant dans Matlab

```
>> [dis,disx,disy]=distance(0,0,1,1) ;
```

On verifiera ensuite que dis = 1:4142 et que disx = disy = 1.